Using SiteMinder Administrative REST API to create OpenID Connect configuration

Warren Barrow
Solution Engineer, Identity Security
Broadcom

May 5, 2020

Introduction

Broadcom's Symantec SiteMinder is an enterprise access management system that authenticates users, validates their authorization, provides single sign-on between applications, and provides application protection against a number of OWASP top 10 vulnerabilities.

For SiteMinder administrators, the task of manually managing application policies presents an opportunity for error to be introduced. Customers who wish to integrate application policy protection into their continuous integration / continuous delivery lifecycle can take advantage of the REST API to manage administrative functions. Programmatic updates may be applied during the CI/CD process of onboarding or updating application protectivity.

While the documentation is adequate, it's helpful to be guided through the process to understand the minimum requirements to get started with testing in your own environment. This document's purpose is to guide the administrator through interacting with the REST API to configure the SiteMinder OpenID Connect Provider and an OpenID Connect client application.

Pre-requisites

This document assumes you've set up the initial SiteMinder 12.8.3 environment and prerequisites for the OpenID Connect Provider such as configuring the Session Store and securing the Authentication URL. See Configure SiteMinder as OpenID Connect Provider and Manage OIDC Objects Using REST APIs. Communities Article: CA SSO OpenID Connect Provider

To summarize, you'll need to:

- Configure & enable a Session Store
- Add your federation agent or Access Gateway to the FederationWebServicesAgentGroup object
- Edit the FederationWebServicesRealm
    - Modify Authentication Scheme as needed
    - Create new Rule with Resource Filter /affwebservices/secure/securedirec*
        - Actions: GET/POST
- Create new Realm in FederationWebServicesDomain with Resource Filter /affwebservices/CASSO/oidc/
    - Default Resource Protection: Unprotected

Authentication Token

First step is to obtain an authentication token to be used for subsequent API calls. All API requests are sent to the Policy Server Management UI host. The CURL command below is shown for discussion and may be used on the command line or imported into a REST API Client. A Postman Collection of the APIs calls are available and detailed below.

curl -k --location --request POST 'https://<PolicyServerMgmtUI_IP>:8443/ca/api/sso/services/login/v1/token' --header 'Authorization: Basic Base64EncodedSiteMinderUsernamePassword'

Response:


```
{
    "sessionkey": "eyJlbmMiOiJBMTI4R0NNIiwiYWxnIjoiZGlyIn0...2i029C0Li5GMMtTwXfv8oQ"
}
```


The session key provided in the response is valid for 15 minutes

Create SiteMinder OpenID Connect Provider

You'll need to update some of the variables below to reflect your own environment. If your certificate alias or user directory name has a space in it, you'll need to substitute the space with a plus (+) sign.

- AuthenticationURL - your Access Gateway with Federation enabled
- SigningAlias Path - use a valid certificate's alias name
- (Object) Name
- AuthorizationServerBaseURL
- UserDirectoriesLink Path

```
curl --request POST \
  --url https://<PolicyServerMgmtUI_IP>:8443/ca/api/sso/services/policy/v1/FedOIDCAdminConfigs/ \
  --header 'authorization: Bearer eyJlbmMiOiJBMTI4R0NNIiwiYWxnIjoiZGlyIn0...k1xfBhHG79Vm8S1ujt4zjw' \
  --header 'content-type: application/json' \
  --data '{
"SigningAlgorithm": "RS512",
"AuthenticationURL": "https://AccessGateway/affwebservices/secure/secureredirect",
"ScopeMapping": [
        {
        "ScopeName": "name",
                        "Claims": "name"
                },
                {
                        "ScopeName": "email",
                        "Claims": "email"
                }
        ],
```

```
        "AuthorizationCodeExpiryTime": 60,
        "SigningAlias": {
                "path": "/FedCertificates/YourCertAlias"
        },
        "Name": "SiteMinder_OIDC_Provider",
        "Description": "Created by REST API",
        "AuthorizationServerBaseURL": "https://AccessGateway",
        "UserDirectoriesLink": [
                {
                        "path": "/SmUserDirectories/YourDirectoryName"
                }
        ],
        "ClaimsMapping": [
                {
                        "UserAttribute": "givenname",
                        "Claim": "name"
                },
                {
                        "UserAttribute": "mail",
                        "Claim": "email"
                }
        ],
        "SignIDToken": true,
        "SignUserInfo": false,
        "EncryptIDToken": false,
        "EncryptUserInfo": false,
        "SecureAuthURL": true,
        "AuthenticationType": "LOCAL",
        "MinimumAuthLevel": 5,
        "type": "FedOIDCAdminConfig"
}'
```

Create OpenID Connect Client

This client represents the target application a user will login to after the authentication flow has completed. There are many parameters here to consider based on the configuration of your target application. The example below provides a good starting point and will require your application to know both the ClientID and the ClientSecret. You may choose to omit both of these values and they will be automatically generated and returned in the response. The AdminConfigLink parameter value is dependent upon the name you used in the previous step when creating the OIDC Provider.

```
curl --request POST \
  --url https://<PolicyServerMgmtUI_IP>:8443/ca/api/sso/services/policy/v1/FedOIDCClients/ \
  --header 'accept: application/json' \
  --header 'authorization: Bearer eyJlbmMiOiJBMTI4R0NNIiwiYWxnIjoiZGlyIn0...5QAsilEaHuV2DRxWiFJhYw' \
  --header 'content-type: application/json' \
  --data '{
```

```
"type": "FedOIDCClient",
"Enabled": true,
"Name": "App_Name",
"Description": "App Description",
"ApplicationType": "CONFIDENTIAL",
"EnablePKCESupport": false,
"ClientAuthentication": "CLIENT_SECRET_POST",
"DisableConsentScreen": true,
"Scopes": [
   "openid",
   "name",
   "email"
],
"GrantTypes": [
"authorization_code"
   ],
"ResponseTypes": [
    "Code"
],
"SendIDTokenWithRefreshToken": false,
"PopulateClaimsInIDToken": true,
"RegenerateClientSecretFlag": false,
"IDTokenExpiryTime": 300,
"RefreshTokenExpiry": 86400,
"AccessTokenExpiry": 300,
"PopulateSMSessionInIDToken": false,
"RedirectURI": [
    "https://app_url /protected/redirect"
],
"AdminConfigLink": {
    "path": "/FedOIDCAdminConfigs/SiteMinder_OIDC_Provider"
  }
}'
```

Postman API Collection

The accompanying Postman Collection has variables to set the Policy Server Mgmt UI hostname and maintain the authenticated session across the API calls.

To begin, import the collection and go to Step 1 “Set Hostname”. Select “Tests” and enter your Policy Server Mgmt UI IP address or hostname. Click Send to set the host variable.

Then go to Step 2 “Get Admin Token”, select “Auth”, select “Basic”, enter your SiteMinder admin credentials, and select Send. This will retrieve the Admin API Bearer token and set the sessionkey variable to be used in the additional API requests.

Steps 3 and 4 require a few changes to the JSON request to match your environment. Make those changes and Send the requests. Now verify the additions in the Policy Server Mgmt UI under Federation->OpenID Connect.

OIDC Client Testing

If you do not already have an OIDC client to test with the SiteMinder configuration, an Apache web server with OIDC plugin is easy to configure. The below steps will install Apache, SSL, and the OpenID Connect module.

sudo yum install -y httpd mod_auth_openidc mod_ssl
sudo mkdir /var/www/html/example
sudo vi /var/www/html/example/index.html (This is your test app landing page)

Add the following to the end of the /etc/httpd/conf/httpd.conf file. Substitute your ClientID, ClientSecret, and OIDC URLs configured earlier.

```
OIDCSSLValidateServer Off
OIDCProviderIssuer https://accessgateway-base-url
OIDCClientID
OIDCClientSecret
OIDCProviderAuthorizationEndpoint https://accessgateway/affwebservices/CASSO/oidc/app_name/authorize
OIDCProviderTokenEndpoint https://accessgateway/affwebservices/CASSO/oidc/app_name/token
OIDCRedirectURI http://app_url/example/redirect
OIDCCryptoPassphrase SomePassword
OIDCProviderTokenEndpointAuth client_secret_post
OIDCProviderJwksUri
https://accessgateway/affwebservices/CASSO/oidc/jwks?AuthorizationProvider=SiteMinder_OIDC_Provider
OIDCDefaultURL http://app_url/example/index.html
<Location /example>
AuthType openid-connect
Require valid-user
</Location>
```

Troubleshooting

The following error logs will help in troubleshooting any issues and monitoring the OIDC flow. FWTrace.log on the Access Gateway and smtracedefault.log on the SiteMinder Policy Server will have the most interesting data.

Backing object for OIDC:
You may ignore the following message:
IsAuthorized.cpp:627][CSm_Az_Message::IsAuthorized][oidcp:SSO_OIDC_Provider][][][jsmith][Cannot find Affiliate definition for the agent.]

Redirect URI is missing:
An error in the browser showing “redirect_uri is invalid or missing” means the OIDC Client Redirect URI does not match between the application client config and the OIDC Client config in SiteMinder.

SMASSERTIONREF=QUERY URL redirect error:
A browser redirect that fails on a URL similar to below indicates either the Federation Web Services URL is not protected, an incorrect agent is defined, or the agent is not assigned to the FederationWebServicesAgentGroup object.

https://AccessGateway/affwebservices/secure/secureredirect?SMASSERTIONREF=QUERY&SMASSERTIONREF=QUERY&SMASSERTIONREF=QUERY&SMASSERTIONREF=QUERY&SMASSERTIONREF=QUERY&SMASSERTIONREF=QUERY&SMASSERTIONREF=QUERY&SMASSERTIONREF=QUERY...